```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.text.NumberFormat;
import java.util.Optional;

import javax.swing.plaf.metal.MetalCheckBoxIcon;
import javafx.event.EventHandler;
import javafx.event.ActionEvent;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextInputDialog;
import javafx.scene.control.TableColumn.CellEditEvent;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.Border;
import javafx.scene.layout.BorderStroke;
import javafx.scene.layout.BorderStrokeStyle;
import javafx.scene.layout.BorderWidths;
import javafx.scene.layout.CornerRadii;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.layout.Pane;
import javafx.scene.control.TableView;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableRow;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.control.cell.TextFieldTableCell;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.control.TabPane;
import javafx.scene.control.Tab;
import javafx.scene.layout.GridPane;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.Labeled;
import javafx.scene.control.ListView;
import javafx.scene.control.ContentDisplay;
import javafx.scene.control.cell.ComboBoxTableCell;
import javafx.scene.canvas.*;
import javafx.beans.value.ChangeListener;
```

```
import javafx.beans.value.ObservableValue;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import javafx.scene.layout.FlowPane;
import javafx.scene.layout.BorderPane;
import java.io.File;
import java.io.IOException;
import javafx.embed.swing.SwingFXUtils;
import javafx.scene.image.WritableImage;
import javafx.scene.SnapshotParameters;
import javax.imageio.ImageIO;
import javafx.scene.transform.Transform;
import java.io.FileWriter;
//import java.beans.EventHandler;
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.List;
import javafx.stage.Window;



public class ManageRow3 extends Application{
    private static final int WIDTH = 820;
    private static final int HEIGHT = 600;
    private static final String[] POSITIONS = {"Port", "Starboard", "Both",
"Coxswain"};
    private static final Integer[] BOATS = {1, 2, 4, 8};
    private static final String[] RIGS = {"Port", "Starboard"};

    private Button newBoatButton = new Button("New Boat");

    private static int boatAdded = 0;
    private Boat[] boats = new Boat[20];


    private TextField boatNameField =  new TextField();
    private TextField boatSizeField =  new TextField();
    private TextField rowerNameField =  new TextField();
    private TextField rowerSideField =  new TextField();

    private Canvas boatImg = new Canvas();
    private Canvas lineupsCanvas = new Canvas();


    private BorderPane root = new BorderPane();
    private FlowPane boatInfo = new FlowPane();
    private GridPane boatThumbnails = new GridPane();
    private ScrollPane allThumbnails = new ScrollPane();

    private HBox lineupsTable = new HBox(100);

    private ComboBox boatSizes = new
ComboBox(FXCollections.observableArrayList(BOATS));
```

```
    private ComboBox rowerPosition = new ComboBox();
    private ComboBox rigOptions = new
ComboBox(FXCollections.observableArrayList(RIGS));
    private ComboBox boatsDropDown = new ComboBox();
    private ComboBox ports = new ComboBox();
    private ComboBox starboards = new ComboBox();
    private ComboBox coxs = new ComboBox();

    private TableView<Rower> table;
    TableView<Rower> rowerTable = new TableView<Rower>();
    TableView<Rower> coxTable = new TableView<Rower>();

    private TextArea displayArea =  new TextArea();

    private  ArrayList<Boat> fleet = new ArrayList<Boat>();
    private  ArrayList<Rower> teamRoster = new ArrayList<Rower>();

    private Tab boatsTab = new Tab();
    private Tab lineupsTab = new Tab();
    private Tab rosterTab = new Tab();
    private Tab learnMore = new Tab();

    private Boat currentBoat;

    public void start(Stage stage){
        //create elements for tabs
        //createRowerCombos("roster.csv");
        teamRoster = csvReaderRower("roster.csv");
        fleet = readBoatCsv("boats.csv"); //returns the csv
        buildBoatSelection();
        rowerTable = createRowerRosterView();
        coxTable = createCoxRosterView();

        //create tabs
        TabPane tabPane = new TabPane();
        boatsTab.setText("Boats");
        boatsTab.setClosable(false);
        //Canvas c = new Canvas();
        setBoatTab();



        lineupsTab.setText("Lineups");
        lineupsTab.setClosable(false);
        setLineupsTab();



        rosterTab.setText("Roster");
        rosterTab.setClosable(false);
        setRosterTab();



        learnMore.setText("LearnMore");
        learnMore.setClosable(false);
```

```
        setLearnMoreTab();

        tabPane.getTabs().addAll(boatsTab, lineupsTab, rosterTab, learnMore);
        //create all handlers
        setHandlers();

        //create the scene
        Scene scene = new Scene(tabPane);

        stage.setScene(scene);
        stage.setTitle("Row Manager");
        stage.setWidth(WIDTH);
        stage.setHeight(HEIGHT);

        scene.getStylesheets().add("style.css");
        stage.show();


    }



    public void setBoatTab(){
        HBox boatInfo = new HBox(50);
        VBox name = new VBox(10);
        VBox size = new VBox(10);
        VBox rig = new VBox(10);

        Button saveAndQuit = new Button("Save and Quit");
        HBox saveQuit = new HBox(100);
        saveQuit.getChildren().add(saveAndQuit);
        saveQuit.setAlignment(Pos.TOP_RIGHT);
        saveAndQuit.setOnAction(e-> saveAndQuitHandler());



        Label boatNameLabel = new Label("Boat Name:    ");
        Label boatSizeLabel = new Label("Boat Size:   ");
        Label boatRigLabel = new Label("Boat Rig:   ");

        name.getChildren().addAll(boatNameLabel, boatNameField);
        size.getChildren().addAll(boatSizeLabel, boatSizes);
        rig.getChildren().addAll(boatRigLabel, rigOptions);
        boatInfo.getChildren().addAll(name, size, rig, newBoatButton);

        BorderPane page = new BorderPane();
        Pane wrapperPane = new Pane();

        page.setCenter(wrapperPane);

        // Put canvas in the center of the window
        wrapperPane.getChildren().add(boatImg);
        // Bind the width/height property to the wrapper Pane
        boatImg.widthProperty().bind(wrapperPane.widthProperty());
```

```
        boatImg.heightProperty().bind(wrapperPane.heightProperty());

        page.setPadding(new Insets(10));
        page.setTop(boatInfo);
        page.setBottom(saveQuit);

        popThumbnails();



        allThumbnails.setContent(boatThumbnails);
        page.setRight(allThumbnails);
        boatsTab.setContent(page);
    }

    public void setLineupsTab(){
        boatsDropDown.setPrefWidth(100);

        Button saveAndQuit = new Button("Save and Quit");
        HBox saveQuit = new HBox(100);
        //saveQuit.getChildren().add(saveAndQuit);
        saveQuit.setAlignment(Pos.TOP_RIGHT);
        saveAndQuit.setOnAction(e-> saveAndQuitHandler());

        //System.out.println(fleet.toString());
        //boatsDropDown.getItems().clear();
        // for(Boat b : fleet){ //read the csv here, create combo box

        //     boatsDropDown.getItems().add(b.getName());
        // }
        if(currentBoat != null)
        {
            boatsDropDown.setValue(currentBoat.getName());
        }
        //System.out.println(currentBoat);
        VBox selectBoat = new VBox(20);
        selectBoat.getChildren().addAll(boatsDropDown);

        //make all canvas things
        BorderPane lineupsPane = new BorderPane();
        Pane wrapperPane = new Pane();
        wrapperPane.setPrefWidth(400);
        wrapperPane.setPrefHeight(400);

        //bind the canvas
        wrapperPane.getChildren().addAll(lineupsCanvas);
        lineupsCanvas.widthProperty().bind(wrapperPane.widthProperty());
        lineupsCanvas.heightProperty().bind(wrapperPane.heightProperty());

        // Boat b = new Boat(8, "testdraw", 1);
        // GraphicsContext gc = lineupsCanvas.getGraphicsContext2D();
        // gc.clearRect(0, 0, lineupsCanvas.getWidth(),
lineupsCanvas.getHeight());
        // b.drawBoat(gc);
```

```
        lineupsPane.setPadding(new Insets(10));
        lineupsPane.setTop(selectBoat);
        lineupsPane.setLeft(wrapperPane);

        TableView<Rower> rosterTable = createCoxRowerRosterView();

        HBox test = new HBox(10);
        //this is for the proof of consept
        //Boat b = new Boat(4, "Conte", 1);
        if(currentBoat != null){
            test = lineupsTable(currentBoat);
            lineupsPane.setBottom(test);
            String boatName = String.valueOf(boatsDropDown.getValue());
            GraphicsContext gc = lineupsCanvas.getGraphicsContext2D();
            gc.clearRect(0, 0, lineupsCanvas.getWidth(),
lineupsCanvas.getHeight());
            Boat b =  Boat.getBoat(boatName, fleet);
            b.drawBoat(gc);
        }
        VBox rosterTableHolder = new VBox(10);
        //rosterTableHolder.setPrefWidth(200);
        rosterTableHolder.getChildren().addAll(rosterTable, saveAndQuit);


        lineupsPane.setRight(rosterTableHolder);
        lineupsPane.setBottom(test);

        lineupsTab.setContent(lineupsPane);
    }

    private void setRosterTab(){
        BorderPane rosterPane = new BorderPane();
        VBox rowers = new VBox(10);
        rosterPane.setPadding(new Insets(10));

        Button saveAndQuit = new Button("Save and Quit");
        HBox saveQuit = new HBox(100);
        saveQuit.getChildren().add(saveAndQuit);
        saveQuit.setAlignment(Pos.BOTTOM_RIGHT);
        saveAndQuit.setOnAction(e-> saveAndQuitHandler());


        HBox nameAndWeight = new HBox(10);
        Label rowerName = new Label("Name");
        TextField rowerNameField2 = new TextField();
        Label weight = new Label("Lbs");
        TextField weightField = new TextField();
        nameAndWeight.getChildren().addAll(rowerName, rowerNameField2, weight,
weightField);

        HBox positionAndRemove = new HBox(10);
        Label positionLabel = new Label("Position");
```

```
        ComboBox positionDropDown = new ComboBox();
        for(int i = 0; i < POSITIONS.length-1; i++)
        {
            positionDropDown.getItems().add(POSITIONS[i]);;
        }
        Button removeRowerButton = new Button("Remove");
        positionAndRemove.getChildren().addAll(positionLabel, positionDropDown,
removeRowerButton);

        HBox ergScoreAndSave = new HBox(10);
        Label ergLabel = new Label("2k");
        TextField ergScore = new TextField();
        Button saveRowerButton = new Button("Save");
        ergScoreAndSave.getChildren().addAll(ergLabel, ergScore, saveRowerButton);


        rowers.getChildren().addAll(nameAndWeight, positionAndRemove,
ergScoreAndSave);



        VBox rowerTableBox = new VBox(20);
        VBox coxTableBox = new VBox(20);
        //Create stuff below cox table
        VBox coxFields = new VBox(20);
        HBox coxNameFields = new HBox(10);
        HBox coxyearFields = new HBox(10);

        Label coxNameLabel = new Label("Name");
        TextField coxNameField = new TextField();

        Button removeCoxButton = new Button("Remove");
        removeRowerButton.setOnAction(e-> {
            teamRoster.remove(rowerTable.getSelectionModel().getSelectedItem());
            setRosterTab();
            setLineupsTab();
        });
        removeCoxButton.setOnAction(e-> { //here
            teamRoster.remove(coxTable.getSelectionModel().getSelectedItem());
            setRosterTab();
            setLineupsTab();
        });

        coxNameFields.getChildren().addAll(coxNameLabel, coxNameField,
removeCoxButton);

        Label coxYearLabel = new Label("Year");
        ComboBox yearDropDown = new ComboBox();
        for(int i = 2023; i < 2027; i++)
        {
            yearDropDown.getItems().add(i);;
        }
        Button addCoxButton = new Button("Save");
        coxyearFields.getChildren().addAll(coxYearLabel, yearDropDown,
```

```
addCoxButton);

        coxFields.getChildren().addAll(coxNameFields, coxyearFields);

        rowerTableBox.getChildren().addAll(rowerTable, rowers);
        rosterPane.setLeft(rowerTableBox);

        coxTableBox.getChildren().addAll(coxTable, coxFields);
        rosterPane.setRight(coxTableBox);
        rosterPane.setBottom(saveQuit);

        rosterTab.setContent(rosterPane);

        addCoxButton.setOnAction(e-> addCoxButtonHandler(coxNameField,
yearDropDown));

        saveRowerButton.setOnAction(e-> {
            if(rowerNameField2.getText().equals("")){System.out.println("You must
enter a name to add a Rower"); return;}
            else if(positionDropDown.getValue() == null){System.out.println("You
must select a position to add a Rower"); return;}
            else if(ergScore.getText().equals("")){System.out.println("You must
enter an erg score to add a Rower"); return;}
            else if(weightField.getText().equals("") ||
!isNumeric(weightField.getText())){System.out.println("You must enter a valid
weight to add a Rower"); return;}
            Rower temp = new Rower(rowerNameField2.getText(),
String.valueOf(positionDropDown.getValue()), ergScore.getText(),
Double.valueOf(weightField.getText()));
            teamRoster.add(temp);
            //System.out.println(teamRoster.toString());

            /* createRowerRosterView();
            createCoxRosterView(); */
            setRosterTab();
            setLineupsTab();

        }); //working here
    }

    private boolean isNumeric(String s)
    {
        for(char c : s.toCharArray())
        {
            if(!Character.isDigit(c) && c != '.')
            {
                return false;
            }
        }
        return true;
    }

    private void setHandlers(){
        newBoatButton.setOnAction(e-> addBoat());
```

```
        boatsDropDown.setOnAction(e -> selectBoat()); //make this


    }
    public void setLearnMoreTab(){
        Image gifImage = new Image("LearnMore.gif");

        // Create an ImageView object to display the GIF image
        ImageView gifImageView = new ImageView(gifImage);

        // Create a Pane to hold the ImageView
        ScrollPane pane = new ScrollPane(gifImageView);
        learnMore.setContent(pane);
    }

    //********************* Helpers *********************/
    public void addCoxButtonHandler(TextField coxNameField, ComboBox yearDropDown)
{
        if(coxNameField.getText().equals("")){System.out.println("You must enter a
name to add a Coxswain");return;}
        else if(yearDropDown.getValue() == null){System.out.println("You must
select a class year to add a Coxswain");return;}
        Rower temp = new Rower(coxNameField.getText(),
Integer.parseInt(String.valueOf(yearDropDown.getValue())));
        teamRoster.add(temp);
        setRosterTab();
        setLineupsTab();
    }

    public void saveAndQuitHandler(){
        csvWriterRower(teamRoster);
        csvWriterBoat(fleet);
        Scene scene = boatImg.getScene();
        Window wn = scene.getWindow();
        wn.hide();
    }

    public void buildBoatSelection(){
        for(Boat b : fleet){ //read the csv here, create combo box
            boatsDropDown.getItems().add(b.getName());
        }
    }

    public HBox lineupsTable(Boat b)
    {
        HBox output = new HBox(0);
        for(int i = 0; i < b.getLineup().length; i++)
        {
            VBox seat = new VBox(0);
            VBox rower = new VBox(0);
            VBox combo = new VBox(10);
            combo.setPrefWidth(100);
            combo.setPrefHeight(100);
            Label seatLabel;
```

```
            if(i == 0)
            {
                seatLabel = new Label("Bow");
            }
            else if(i == b.getLineup().length-2 || b.getLineup().length == 2)
            {
                seatLabel = new Label("Stroke");
            }
            else if(i == b.getLineup().length-1)
            {
                seatLabel = new Label("Coxswain");
            }
            else
            {
                seatLabel = new Label("" + (i + 1));
            }

            Label rowerLabel;
            ComboBox rowerLabel2;
            rowerLabel2 = new ComboBox();
            rowerLabel2.getItems().add("Empty");
            int seat2 = i;
            rowerLabel2.setOnAction((e) -> {

if(rowerLabel2.getSelectionModel().getSelectedItem().equals("Empty"))
                {
                    b.removeRowerinSeat(seat2);
                    return;
                }
                for(Rower r2 : teamRoster)
                {

if(r2.getName().equals(rowerLabel2.getSelectionModel().getSelectedItem()))
                    {
                        b.addRower(r2, seat2 + 1);
                        setLineupsTab();
                        break;
                    }
                }

            });


            for(Rower r : teamRoster){
                if(b.getLineup().length == 1 && !r.getSide().equals("Coxswain"))
                {
                    rowerLabel2.getItems().add(r.getName());
                }
                else if(i == b.getLineup().length-1 &&
r.getSide().equals("Coxswain")){
                    rowerLabel2.getItems().add(r.getName());
                }
                else if((r.getSide().equals("Port") || r.getSide().equals("Both")
) && (i % 2 == b.getRig() + 1) && (i != b.getLineup().length-1 ||
```

```
b.getLineup().length < 4)){
                    rowerLabel2.getItems().add(r.getName());
                }
                else if((r.getSide().equals("Starboard") ||
r.getSide().equals("Both") ) && (i % 2 == b.getRig()) && (i !=
b.getLineup().length-1 || b.getLineup().length < 4)){
                    rowerLabel2.getItems().add(r.getName());
                }
            }
            if(b.getLineup()[i] != null){
            rowerLabel = new Label(b.getLineup()[i].getName());
                rowerLabel2.setValue(b.getLineup()[i].getName());
            }
            else{
                rowerLabel = new Label("Empty");
                rowerLabel2.setValue("Empty");
            }
            seat.getChildren().add(seatLabel);
            rower.getChildren().add(rowerLabel2);

            combo.getChildren().addAll(seat, rower);

            output.getChildren().add(combo);
        }
        return output;
    }


    public TableView<Rower> createCoxRowerRosterView(){

        ObservableList<Rower> observable_roster =
FXCollections.observableArrayList(teamRoster);
        ListView<Rower> roster_view = new ListView<Rower>();
        roster_view.setItems(observable_roster);

        TableView<Rower> table = new TableView<Rower>();
        table.setEditable(true);
        table.setItems(observable_roster);

        TableColumn<Rower, String> nameCol = new TableColumn<Rower, String>
("Name");
        nameCol.setCellValueFactory(new PropertyValueFactory("name"));
        nameCol.setPrefWidth(100);
        nameCol.setCellFactory(TextFieldTableCell.forTableColumn());
        nameCol.setOnEditCommit(
            new EventHandler<CellEditEvent<Rower, String>>(){
                @Override
                public void handle(CellEditEvent<Rower, String> t) {
                    ((Rower)
t.getTableView().getItems().get(t.getTablePosition().getRow())).setName(t.getNewVa
lue());
                    setRosterTab();
                }
            }
```

```
        );

        TableColumn<Rower, String> sideCol = new TableColumn<Rower, String>
("Position");
        sideCol.setCellValueFactory(new PropertyValueFactory("side"));
        sideCol.setPrefWidth(100);
        sideCol.setCellFactory(TextFieldTableCell.forTableColumn());
        sideCol.setOnEditCommit(
            new EventHandler<CellEditEvent<Rower, String>>(){
                @Override
                public void handle(CellEditEvent<Rower, String> t) {
                    ((Rower)
t.getTableView().getItems().get(t.getTablePosition().getRow())).setSide(t.getNewVa
lue());
                    setRosterTab();
                }
            }
        );

        TableColumn<Rower, String> ergCol = new TableColumn<Rower, String>("2k");
        ergCol.setCellValueFactory(new PropertyValueFactory("ergScore"));
        ergCol.setPrefWidth(100);
        ergCol.setCellFactory(TextFieldTableCell.forTableColumn());
        ergCol.setOnEditCommit(
            new EventHandler<CellEditEvent<Rower, String>>(){
                @Override
                public void handle(CellEditEvent<Rower, String> t) {
                    ((Rower)
t.getTableView().getItems().get(t.getTablePosition().getRow())).setErgScore(t.getN
ewValue());
                    setRosterTab();
                }
            }
        );


        table.getColumns().setAll(nameCol, sideCol, ergCol);

        return table;
    }
    public TableView<Rower> createRowerRosterView(){
        ArrayList<Rower> rowersOnly = new ArrayList<Rower>();
        for(Rower r: teamRoster)
        {
            if(!r.getSide().equals(POSITIONS[3]))
            {
                rowersOnly.add(r);
            }
        }
        //System.out.println(rowersOnly);
        ObservableList<Rower> observable_roster =
FXCollections.observableArrayList(rowersOnly);
        ListView<Rower> roster_view = new ListView<Rower>();
        roster_view.setItems(observable_roster);
```

```
        TableView<Rower> table = new TableView<Rower>();
        table.setItems(observable_roster);
        table.setEditable(true);

        TableColumn<Rower, String> nameCol = new TableColumn<Rower, String>
("Name");
        nameCol.setCellValueFactory(new PropertyValueFactory("name"));
        nameCol.setPrefWidth(100);
        nameCol.setCellFactory(TextFieldTableCell.forTableColumn());
        nameCol.setOnEditCommit(
            new EventHandler<CellEditEvent<Rower, String>>(){
                @Override
                public void handle(CellEditEvent<Rower, String> t) {
                    ((Rower)
t.getTableView().getItems().get(t.getTablePosition().getRow())).setName(t.getNewVa
lue());
                    setLineupsTab();
                }
            }
        );

        TableColumn<Rower, String> sideCol = new TableColumn<Rower, String>
("Position");
        sideCol.setCellValueFactory(new PropertyValueFactory("side"));
        sideCol.setPrefWidth(100);
        sideCol.setCellFactory(TextFieldTableCell.forTableColumn());
        sideCol.setOnEditCommit(
            new EventHandler<CellEditEvent<Rower, String>>(){
                @Override
                public void handle(CellEditEvent<Rower, String> t) {
                    ((Rower)
t.getTableView().getItems().get(t.getTablePosition().getRow())).setSide(t.getNewVa
lue());
                    setLineupsTab();
                }
            }
        );

        TableColumn<Rower, String> ergCol = new TableColumn<Rower, String>("2k");
        ergCol.setCellValueFactory(new PropertyValueFactory("ergScore"));
        ergCol.setPrefWidth(100);
        ergCol.setCellFactory(TextFieldTableCell.forTableColumn());
        ergCol.setOnEditCommit(
            new EventHandler<CellEditEvent<Rower, String>>(){
                @Override
                public void handle(CellEditEvent<Rower, String> t) {
                    ((Rower)
t.getTableView().getItems().get(t.getTablePosition().getRow())).setErgScore(t.getN
ewValue());
                    setLineupsTab();
                }
            }
        );
```

```
        TableColumn<Rower, Double> weightCol = new TableColumn<Rower, Double>
("Weight");
        weightCol.setCellValueFactory(new PropertyValueFactory("weight"));
        weightCol.setPrefWidth(100);

        table.getColumns().setAll(nameCol, sideCol, ergCol, weightCol);

        //csvWriterRower(rowersOnly);
        return table;
    }

    public TableView<Rower> createCoxRosterView(){
        ArrayList<Rower> coxesOnly = new ArrayList<Rower>();
        for(Rower r: teamRoster)
        {
            if(r.getSide().equals(POSITIONS[3]))
            {
                coxesOnly.add(r);
            }
        }
        ObservableList<Rower> observable_roster =
FXCollections.observableArrayList(coxesOnly);
        ListView<Rower> roster_view = new ListView<Rower>();
        roster_view.setItems(observable_roster);

        TableView<Rower> table = new TableView<Rower>();
        table.setEditable(true);
        table.setItems(observable_roster);

        TableColumn<Rower, String> nameCol = new TableColumn<Rower, String>
("Name");
        nameCol.setCellValueFactory(new PropertyValueFactory("name"));
        nameCol.setPrefWidth(150);
        nameCol.setCellFactory(TextFieldTableCell.forTableColumn());
        nameCol.setOnEditCommit(
            new EventHandler<CellEditEvent<Rower, String>>(){
                @Override
                public void handle(CellEditEvent<Rower, String> t) {
                    ((Rower)
t.getTableView().getItems().get(t.getTablePosition().getRow())).setName(t.getNewVa
lue());
                    setLineupsTab();
                }
            }
        );

        TableColumn<Rower, String> classCol = new TableColumn<Rower, String>
("Year");
        classCol.setCellValueFactory(new PropertyValueFactory("classYear"));
        classCol.setPrefWidth(150);


        table.getColumns().setAll(nameCol, classCol);
```

```
        return table;
    }


    public void saveImg(Canvas canvas, String name){
        WritableImage writableImage = new WritableImage(400, 400);
        SnapshotParameters params = new SnapshotParameters();
        params.setTransform(Transform.scale(1, 1)); // double the scale factor
        canvas.snapshot(params, writableImage);
        // = "test3";
        String test = name.concat(".png");
        System.out.println(test);
        File file = new File(test);
        System.out.println("in save img");

        try{
            ImageIO.write(SwingFXUtils.fromFXImage(writableImage, null), "png",
file);
        }
        catch (IOException e){
            System.out.println("didn't save");
        }
        System.out.println("exiting save img");

    }

    //not used rn
    public void createRowerCombos(String filePath){
        teamRoster = csvReaderRower(filePath);
        for(Rower r : teamRoster){
            if(r.getSide().equals("Coxswain")){
                coxs.getItems().add(r.getName());
            }
            else if((r.getSide().equals("Port") || r.getSide().equals("Both") )){
                ports.getItems().add(r.getName());
            }
            else if((r.getSide().equals("Starboard") ||
r.getSide().equals("Both"))){
                starboards.getItems().add(r.getName());
            }
        }
    }

    public void popThumbnails() { //throws FileNotFoundException{

        int toBoatAdded = 0;
        int debug = 0;
     //   while (toBoatAdded == boatAdded) {

    //    }

        for(int i = 0; i < 2; i ++){
            for(int j = 0; j < 8; j ++){
```

```
              //  Button b = new Button("Boat Here" + debug++);
                Canvas c = new Canvas(100, 100);

                if (toBoatAdded < boatAdded) {
                    GraphicsContext gc = c.getGraphicsContext2D();
                    boats[toBoatAdded++].drawBoat(gc);
                    System.out.println("called " + boatAdded);
                }


              //  b.setPadding(new Insets(10));
                boatThumbnails.add(c, i, j);

             //   Image img = new Image(new FileInputStream("testimg4.png"), 100,
100, true, false);
              //   ImageView imgIcon = new ImageView(img);

            //    b.setGraphic(imgIcon);
            }
        }
    }

    // public void popThumbnails() throws FileNotFoundException{
    //     for(int i = 0; i < 2; i ++){
    //         for(int j = 0; j < 8; j ++){
    //             Button b = new Button("Boat Here");
    //             b.setPadding(new Insets(10));
    //             boatThumbnails.add(b, i, j);
    //             Image img = new Image(new FileInputStream("testimg4.png"), 100,
100, true, false);
    //             ImageView imgIcon = new ImageView(img);
    //             b.setGraphic(imgIcon);
    //         }
    //     }
    // }

    // public HBox lineupsTable(Boat b){
    //     HBox lineup = new HBox(10);
    //     for(int i = 0; i < b.getLineup().length; i++){
    //         int rig = b.getRig();
    //         VBox seat = new VBox(0);
    //         VBox rower = new VBox(0);
    //         VBox combo = new VBox(10);
    //         combo.setPrefWidth(100);
    //         combo.setPrefHeight(100);
    //         Label seatLabel;
    //         if(i == 0){
    //             seatLabel = new Label("Bow");
    //         }
    //         else if(i == b.getLineup().length-2){
    //             seatLabel = new Label("Stroke");
    //         }
    //         else if(2% i == 1){
```

```
    //             rower.getChildren().add(starboards);
    //         }
    //         else if(2% i == 0){
    //             rower.getChildren().add(ports);
    //         }
    //         else if(i == b.getLineup().length-1){
    //             seatLabel = new Label("Coxswain");
    //             rower.getChildren().add(coxs);
    //         }
    //         else{
    //             seatLabel = new Label("" + (i + 1));
    //         }
    //         combo.getChildren().addAll(seatLabel, rower);
    //         lineup.getChildren().add(combo);
    //     }
    //     return lineup;

    // }

    //********************* Handlers **********************/
    public void addBoat(){
        //check rig to draw boat
        int rigin = 0;
        if(String.valueOf(rigOptions.getValue()) == RIGS[1]){
            rigin = 1;
        }
        //for the CSV
        String boatName = boatNameField.getText();


        Boat b = new Boat(Integer.valueOf(String.valueOf((boatSizes.getValue()))),
boatNameField.getText(), rigin);
        boats[boatAdded++] = b;

        fleet.add(b);
        csvWriterBoat(fleet); //ideally this would only do new ones but we dont
have a save button
        boatsDropDown.getItems().add(b.getName());
        //draw the boat
        GraphicsContext gc = boatImg.getGraphicsContext2D();
        gc.clearRect(0, 0, boatImg.getWidth(), boatImg.getHeight());

        b.drawBoat(gc);
        //saveImg(c, boatName);

        //popThumbnails(boatName);
        //reset the page
        boatSizeField.setText("");
        boatNameField.setText("");

        popThumbnails();

    }
    public void selectBoat(){ //only draws boats that have been pre drawn
```

```
        String boatName = String.valueOf(boatsDropDown.getValue());
        //GraphicsContext gc = lineupsCanvas.getGraphicsContext2D();
        Boat b =  Boat.getBoat(boatName, fleet);
        //b.drawBoat(gc, -1);
        //b.drawBoat(gc, -1);
        lineupsTable = lineupsTable(b);
        currentBoat = b;
        setLineupsTab();
        GraphicsContext gc = lineupsCanvas.getGraphicsContext2D();
        gc.clearRect(0, 0, lineupsCanvas.getWidth(), lineupsCanvas.getHeight());
        b.drawBoat(gc);
    }

    //********************** CSV Tools **********************/
    public void csvWriterRower(ArrayList<Rower> data) {
        //ArrayList<String> data = new ArrayList<String>();

        String csvFilePath = "roster.csv";
        FileWriter csvWriter = null;
        try {
            csvWriter = new FileWriter(csvFilePath);

            for (Rower line : data) {
                if(line.getSide() == "Port" || line.getSide() == "Starboard" ||
line.getSide() == "Both"){
                    csvWriter.append(line.getName());
                    csvWriter.append("|");
                    csvWriter.append(line.getSide());
                    csvWriter.append("|");
                    csvWriter.append(String.valueOf(line.getWeight()));
                    csvWriter.append("|");
                    csvWriter.append(String.valueOf(line.getErgScore()));
                    csvWriter.append("\n");

                }
                else{
                    csvWriter.append(line.getName());
                    csvWriter.append("|");
                    csvWriter.append(line.getSide());
                    csvWriter.append("|");
                    csvWriter.append(String.valueOf(line.getWeight()));
                    csvWriter.append("|");
                    csvWriter.append(String.valueOf(line.getClassYear()));
                    csvWriter.append("\n");
                }
            }
            csvWriter.flush();
        } // changing csv writers to reflect the cox position needs


        catch (IOException e) {
            e.printStackTrace();
        }
         try {
```

```
            csvWriter.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }

    public ArrayList<Rower> csvReaderRower(String filePath) {
        ArrayList<Rower> dataList = new ArrayList<Rower>();
        try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
            String line;
            Rower data;
            while ((line = br.readLine()) != null) {
                //System.out.println(line);
                String[] values = line.split("\\|");
                //System.out.println(values[0]);
                if(values[1].equals("Coxswain")){
                    data = new Rower(values[0], Integer.valueOf(values[3]));

                }
                else{
                    data = new Rower(values[0], values[1], values[3],
Double.valueOf(values[2])); // assuming the CSV has three columns
                }
                //name, size, rig
                //System.out.println(data);
                dataList.add(data);
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }

        return dataList;
    }


    public void csvWriterBoat(ArrayList<Boat> data) {
        //ArrayList<String> data = new ArrayList<String>();

        String csvFilePath = "boats.csv";
        FileWriter csvWriter = null;

        try {
            csvWriter = new FileWriter(csvFilePath);
            for (Boat line : data) {
                csvWriter.append(line.getName());
                csvWriter.append("|");
                csvWriter.append(String.valueOf(line.getSizeForWriting()));
                csvWriter.append("|");
                csvWriter.append(String.valueOf(line.getRig()));
                csvWriter.append("\n");

            }
```

```
            csvWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                csvWriter.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

        public static ArrayList<Boat> readBoatCsv(String filePath) {
            ArrayList<Boat> dataList = new ArrayList<Boat>();
            try (BufferedReader br = new BufferedReader(new FileReader(filePath)))
{
                String line;
                while ((line = br.readLine()) != null) {
                    //System.out.println(line);
                    String[] values = line.split("\\|");
                    //System.out.println(values[0]);
                    Boat data = new Boat(Integer.valueOf(values[1]), values[0],
Integer.valueOf(values[2])); // assuming the CSV has three columns
                    //name, size, rig
                    //System.out.println(data);
                    dataList.add(data);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
            return dataList;
        }
    }
```